DRAFT PCI-SIG ENGINEERING CHANGE NOTICE

TITLE:	Expansion ROM Validation
DATE:	Initial: Jan 26, 2017 Cross Workgroup Review: March 9, 2017 Member Review: April 13. 2017
AFFECTED DOCUMENT:	PCI Express Base Specification 4.0 Revision 0.7
SPONSOR:	Steve Glaser, Nvidia, Eric N. Lais, IBM

Summary of the Functional Changes

Provide an optional mechanism to indicate to software the results of a hardware validation of Expansion ROM contents.

Cleanup of text Expansion ROM text is part of this ECN. Field definitions are moved into a new table after the existing figure. This follows existing PCIe convention and makes this ECN easier to write. This cleanup may instead be implemented as part of the PCIe Base 4.0 development process.

This ECN is written against the 0.7 Revision of the Draft Base 4.0 specification. The technical changes are compatible with Base 3.1a systems, but the ECN would need to be rewritten to target the Conventional PCI Specification.

Benefits as a Result of the Changes

Defines a mechanism for reporting the results of implementation specific validation of ROM contents. This report is advisory in nature and does not affect access to the expansion ROM.

Many existing implementations support ROM validation. These systems are not affected.

This ECN defines a standardized mechanism to report validation results. This in turn enables a variety of optional generic software mechanisms such as:

1. Delay using expansion ROMs until they pass validation
2. Generate validation status reports
3. Flag warning / needs attention status

Validation status provides a Pass/Fail indication with a small number of optional variations of Pass and Fail. These include:

- Pass Complete success
- Pass with Warning "Certificate expiring soon", etc.
- Fail Invalid "I've been hacked"

- Fail Untrusted "Internally consistent but unsigned"
- Fail Invalid 2 "I've been hacked" with Implementation Specific "condition"
- Fail Untrusted 2 "Internally consistent but unsigned" with Implementation Specific "condition".

Validation is permitted to include additional internal information (e.g. internal firmware).

Assessment of the Impact

Optional normative behavior. No impact if the feature is not implemented.

Analysis of the Hardware Implications

Optional normative behavior. No impact if the feature is not implemented.

Analysis of the Software Implications

Optional normative behavior. New software can wait for validation to complete. Existing software is not affected.

Analysis of the C&I Test Implications

No C&I Tests of the validation mechanism itself are envisioned. The usual register bit read/write testing should be added.

Analysis of the Virtualization Implications

This mechanism is part of the PF. This field, like the rest of the Expansion ROM Base Address Register is not implemented in the VF.

SR-IOV is not affected.

PCIe Base 0.7 Section 10.3.4.1.15 is not affected.

Detailed Description of the change

Change Section 7.5.2.4 as follows:

7.5.2.4 Expansion ROM Base Address Register (Offset 30h)

Some Functions, especially those that are intended for use on add-in cards, require local EPROMs for expansion ROM (refer to the *PCI Firmware Specification* for a definition of ROM contents). This register is defined to handle the base address and size information for this expansion ROM. The register layout is shown in Figure 7-13 and Table 7-x describes the bits in the register. ~~shows how this register is organized.~~

~~The register functions exactly like a 32-bit Base Address register except that the encoding (and usage) of the bottom bits is different. The upper 21 bits correspond to the upper 21 bits of the Expansion ROM base address. The number of bits (out of these 21) that a Function actually implements depends on how much address space the Function requires. For instance, a Function that requires a 64 KB area to map its expansion ROM would implement the top 16 bits in the register, leaving the bottom 5 (out of these 21) hardwired to 0b. Functions that support an expansion ROM must implement this register.~~

~~Device independent configuration software can determine how much address space the Function requires by writing a value of all 1's to the address portion of the register and then reading the value back. The Function will return 0's in all don't-care bits, effectively specifying the size and alignment requirements. The amount of address space a Function requests must not be greater than 16 MB.~~

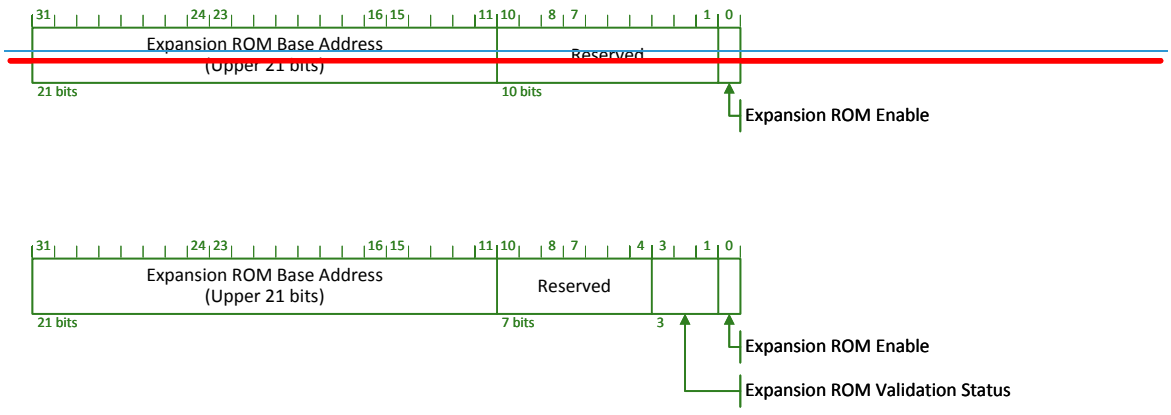

Figure7-13: Expansion ROM Base Address Register Layout

Table 7-x: Expansion ROM Base Address Register

Bit Location	Description	Attributes
0	**Expansion ROM Enable** – This bit ~~Bit 0 in the register is used to~~ controls whether or not the Function accepts accesses to its expansion ROM. Functions that support an Expansion ROM must implement this bit. Functions that do not support an Expansion ROM are permitted to hardwire this bit to 0b. When this bit is 0b,	RO/RW

Bit Location	Description	Attributes
	the Function's expansion ROM address space is disabled. When the bit is 1b, address decoding is enabled using the Expansion ROM Base Address field in this ~~parameters in the other part of the Expansion ROM Base Address~~ register. This allows a Function to be used with or without an expansion ROM depending on system configuration. The Memory Space Enable bit in the Command register has precedence over the Expansion ROM Enable bit. A Function must claim accesses to its expansion ROM only if both the Memory Space Enable bit and the Expansion ROM Enable bit are Set. The default value of this bit is 0b. In order to minimize the number of address decoders needed, a Function may share a decoder between the Expansion ROM Base Address register and other Base Address registers or entry in the Enhanced Allocation capability[1]. When ~~expansion~~ Expansion ROM ~~decode~~Enable is ~~enabled~~Set, the decoder is used for accesses to the expansion ROM and device independent software must not access the Function through any other Base Address registers or entry in the Enhanced Allocation capability. If the Function has an Enhanced Allocation Extended Capability that includes an EA entry for an Expansion ROM, the Function must not share address decoding resources and this bit must be hardwired to 0b (see Section 7.9.2).	
3:1	**Expansion ROM Validation Status** – Expansion ROM Validation is optional. When this field is non-zero, it indicates the status of hardware validation of the Expansion ROM contents. • An Expansion ROM is considered valid if it passes an Implementation Specific integrity check. • A valid Expansion ROM is also considered trusted if passes an Implementation Specific trust test (e.g. signed by a trusted certificate). • A valid and trusted Expansion ROM can also have an Inplementation Specific warning status (e.g. certificate expiring soon). • Hardware validation must include the contents of the Expansion ROM. This validation status is also permitted to cover additional internal information (e.g., internal firmware). Validation does not include Vital Product Data (see Section TBD). • It is optional whether an implementation is capable of returning Validation Status values 011b, 101b, 110b, or 111b. Values of this field are: **Value Meaning** 000b No Validation Validation not supported 001b Validation in Progress	HwInit/RO

[1] Note that it is the address decoder that is shared, not the registers themselves. The Expansion ROM Base Address register and other Base Address registers or entries in the Enhanced Allocation capability must be able to hold unique values at the same time.

Bit Location	Description	Attributes
	010b Validation Passed Valid and trusted contents 011b Validation Passed Valid and trusted contents with Implementation Specific warning 100b Validation Failed Invalid contents, untrusted 101b Validation Failed Valid but untrusted contents (e.g., Out of Date, Expired or Revoked Certificate) 110b Validation Failed Invalid contents, untrusted Implementation Specific Reason 111b Validation Failed Valid but untrusted contents Implementation Specific Reason • If the Function does not support validation, this field is hardwired to 000b. • If the Function supports validation and has an Enhanced Allocation Extended Capability with an EA entry for an Expansion ROM, this field is HwInit and its value must be between 010b and 111b (see Section 7.9.2). • Otherwise, this field is Read Only and has a default value of 001b. When validation completes, this field must contain a value between 010b and 111b. Software is permitted to assume validation will never complete if this field contains 001b and 1 minute has passed after deassertion of Fundamental Reset. This field is not affected by resets other than Fundamental Reset.	
10:4	**Reserved**	RsvdP
31:11	**Expansion ROM Base Address** – contains the upper bits of the starting address of the Expansion ROM. ~~The register~~ This field functions exactly like the address portion of a 32-bit Base Address register ~~except that the encoding (and usage) of the bottom bits is different~~. ~~The upper 21 bits~~ This field corresponds to the upper 21 bits of the Expansion ROM base address. The number of bits (out of these 21) that a Function actually implements depends on how much address space the Function requires. For instance, a Function that requires a 64 KB area to map its expansion ROM would implement the top 16 bits in ~~the register~~this field, leaving the bottom 5 (out of these 21) hardwired to 0b. Functions that support an expansion ROM must implement this ~~register~~field. Functions that do not support an expansion ROM are permitted to hardwire this field to 0. Device independent configuration software can determine how much address space the Function requires by writing a value of all 1's to this field ~~the address portion of the register~~ and then reading the value back. The Function will return 0's in all don't-care bits, effectively specifying the size and alignment requirements. The amount of address space a Function requests must not be greater than 16 MB. If the Function has an Enhanced Allocation Extended Capability that includes an EA entry for an Expansion ROM, this field must be hardwired to 0 (see Section 7.9.2).	RW

Change Section 6.24, Enhanced Allocation, as follows:

A Function with an Expansion ROM is permitted use the existing mechanism or the EA mechanism, but is not permitted to support both. If a Function uses the EA mechanism (EA entry with BEI of 8), the Expansion ROM Base Address and Expansion ROM Enable fields ~~Register (offset 30h)~~ must be hardwired to 0 (see Section 7.5.2.4). The Enable bit of the EA entry is equivalent to the Expansion ROM Enable bit. If a Function uses Expansion ROM Base Address Register mechanism , no EA entry with a BEI of 8 is permitted. In both cases, Firmware Validation, if supported, uses the Expansion ROM Validation Status field (see Section 7.5.2.4)

Change Section 7.9.2, Enhanced Allocation Per-Entry Format, as follows:

- ❑ At most one entry is permitted with a BEI of 8; If such an entry is present, behavior of the Expansion ROM Base Address Register is changed (see Section 7.5.2.4)~~must be hardwired to 0~~.